Documentation API Eurheka

Note : (*) : Requêtes soumises à une authentification de l'utilisateur

Chemin de l'API:
/api : point d'entrée de l'API

Nota : Il est nécessaire de créer les répertoires suivants dans le back :
uploads/
- /cvs
- /docs
- /offers
- /jobs

Documentation API Eurheka

- Gestion des Ressources : /api/resource/
  - (*)GET : /api/resource/bycat/:id
  - (*)GET : api/adminCat/:id
  - (*)GET : api/admin/:id
  - (*)POST : /api/resource/job/ , /api/resource/doc
  - (*)POST : /api/resource/video
  - (*)PUT : /api/resource/:id
  - (*)DELETE : /api/resource/:id
- Gestion des événements : /api/event/
  - (*)GET : /api/event/myevents/
  - (*)GET : /api/event/admin/
  - GET : /api/event/nextEvent
  - (*)GET : /api/event/myRDV
  - (*)GET : /api/event/:id
  - (*)POST : /api/event/
  - (*)PUT : /api/event/rdv/:id
  - (*)DELETE : /api/event/:id
  - Catégories des évènements : /api/event/category/
    - GET : /api/event/category/
    - GET : /api/event/category/:id
    - (*)POST : /api/event/category/
    - (*)PUT : /api/event/category/:id
    - (*)DELETE : /api/event/category/:id
- Authentification : /api/session/
  - Get /api/session :
- CV : /api/cv/
  - (*)GET /api/cv/
  - (*)GET /api/cv/admin/
  - (*)POST /api/cv/
  - (*)DELETE /api/cv/:id
- Entreprise : /api/enterprise/
  - GET /api/enterprise/
  - GET /api/enterprise/:id
  - POST /api/enterprise/
  - PUT /api/enterprise/:id
  - DELETE /api/enterprise/:id
- Offres d’emploi : /api/job/
  - GET /api/job/
  - (*)POST /api/job/
  - (*)DELETE /api/job/:id
  - GET /api/job/offertype/
  - GET /api/job/count/
  - (*)GET /api/job/admin/

GET /api/job/category/

Avis : /api/opinion/

(*) GET /api/opinion/

GET /api/opinion/eurheka/

(*) GET /api/opinion/:id

(*) GET /api/opinion/enterprise/:id

(*)POST /api/opinion/

(*) PUT /api/opinion/:id

(*) DELETE /api/opinion/:id

Thèmes : /api/theme/

GET /api/theme/

GET /api/theme/:id

(*)POST /api/theme/

(*)PUT /api/theme/:id

(*)DELETE /api/theme/:id

GET /api/theme/admin/

Utilisateurs : /api/users/

(*)GET /api/users/admin/

(*)GET /api/users/:id

POST /api/users/

(*)PUT /api/users/:id

(*)PUT /api/users/admin/:id

(*)DELETE /api/users/:id

POST /api/users/login/

GET /api/users/logout/

POST /api/users/lost/

POST /api/users/newPass/

Envoi d’email : /api/mail/

POST /api/mail/

Gestion des Ressources : /api/resource/

(*)GET : /api/resource/bycat/:id

Récupère les ressources pour une catégorie, et en fonction de l'utilisateur

Entrées :

id : Id de la catégorie.(1 : vidéo, 2 documents à télécharger, 3 fiches métiers),
token pour le niveau de l'utilisateur

Sortie :

Tableau d'objets Json représentant les ressources dispo
(http 200 OK, 500 erreur interne)


```
[
  {
    "id_resource": 12,
    "id_cat": 1,
    "name": "name",
    "path": "path_to_doc"
  },
]
```


(*)GET : api/adminCat/:id

Récupère toutes les ressources pour une catégorie

Entrées :

id : Id de la catégorie.(1 : vidéo, 2 documents à télécharger, 3 fiches métiers)

Sortie :

Tableau d'objets Json représentant les ressources dispo
(http 200 OK, 500 erreur interne)


```
[
  {
    "id_resource": 12,
    "id_cat": 1,
    "name": "name",
    "visibility": 1,
    "path": "path_to_doc"
  },
]
```


(*)GET : api/admin/:id

Récupère une ressource par son ID

Entrées :

id : Id de la ressource

Sortie :

objet JSON représentant la ressource, ainsi que les thèmes (sous forme de tableau)
(http 200 OK, 500 erreur interne)

```
{
  "name": "docname",
  "idDoc": 11,
  "path": "path_to_doc",
  "idCat": 2,
  "visibility": 1,
  "CategoryResource": "Documents à télécharger",
  "themes": [
    {
      "idTheme": 25,
      "checked": false,
      "themeName": "a"
    },
    {
      "idTheme": 26,
      "checked": false,
      "themeName": "theme_name"
    }
]
}
```


(*)POST : /api/resource/job/ , /api/resource/doc

Entrées :

- visibility : entier, indique le niveau de visibilité (utilisateur non connecté, connecté ou entreprise
- name : nom de la ressource
- id_cat : identifiant de la catégorie de la ressource
- file : fichier à télécharger
- themes : tableau contenant les thèmes associés (sous forme d'objet : {id_theme, checked, themeName}

Sortie : 201 OK, 422 erreur de saisie, 500 erreur interne

(*)POST : /api/resource/video

Entrées :

- visibility : entier, indique le niveau de visibilité (utilisateur non connecté, connecté ou entreprise
- name : nom de la ressource
- id_cat : identifiant de la catégorie de la ressource
- video : chemin vers la vidéo (prévue pour Youtube
- themes : tableau contenant les thèmes associés (sous forme d'objet : {id_theme, checked, themeName}

Sortie : 201 OK, 422 erreur de saisie, 500 erreur interne

(*)PUT : /api/resource/:id

Entrées :

- visibility : entier, indique le niveau de visibilité (utilisateur non connecté, connecté ou entreprise
- name : nom de la ressource
- themes : tableau contenant les thèmes associés (sous forme d'objet : {id_theme, checked, themeName}

Sortie : 200 OK, 422 erreur de saisie, 500 erreur interne, 404 ressource non trouvée

(*)DELETE : /api/resource/:id

Supprime une ressource par son ID

Entrée :

id de la ressource à supprimer

Sorties :

message d’information

(http 200 OK, 500 erreur, 404 : ressource non trouvée)

Gestion des événements : /api/event/

(*)GET : /api/event/myevents/

Accède aux événements rattachés à un user

Entrées : aucune. id fourni par le token directement en back

Sortie : Tableau d'objets Json représentant les évènements rattachés à l'utilisateur
(401 (unauthorized) 404 (non trouvé) http 200 OK, 500 erreur interne)


```
[
  {
    "eventid": 2,
    "name": "Meeting",
    "event": "2022-03-02T15:12:17.000Z",
    "idCategorie": 1,
    "cat": "RDV",
    "is_owner": 0,
    "id_users": 11,
    "is_valid": 0,
    "date_event": "02/03/2022 15:12"
  }
]
```


(*)GET : /api/event/admin/

Accède aux événements rattachés à un user

Entrées : none

Sortie : Tableau d'objets Json représentant les événements non rattachés à des RDV
(401 (unauthorized) http 200 OK, 500 erreur interne)


```
[
  {
    "id_event": 5,
    "name": "Séminaire Web",
    "date_event": "2022-04-07T16:34:00.000Z",
    "category_name": "Web Atelier",
    "isPassed": false
  },
]
```


GET : /api/event/nextEvent

Accède aux événements à venir qui ne sont pas des RDV

Sortie :

Objet Json représentant le prochain événement
(http 200 OK, 500 erreur interne)


```
{
  "id_event": 5,
  "name": "Séminaire Web",
  "date_eventFR": "07/04/2022 16:34",
  "category_name": "Web Atelier"
}
```


(*)GET : /api/event/myRDV

Accède aux demandes de RDV

Sortie :

Tableau d'objets Json représentant les évènements
(http 200 OK, 500 erreur interne)

```
[
  {
    "eventid": 1,
    "id_users": 7,
    "name": "Demain Avec toto",
    "date_event": "02/03/2022",
    "hour_event": "15:12",
    "is_valid": 1,
    "firstname": "firstname",
    "lastname": "lastname"
  },
]
```


(*)GET : /api/event/:id

Accède à un événement par son id

Entrées : id

Sortie : Objet Json représentant les événements rattachés à l'utilisateur
(401 (unauthorized) 404 (non trouvé) http 200 OK, 500 erreur interne)


```
{
  "id_event": 1,
  "id_cat": 1,
  "name": "Demain Avec toto",
  "date_event": "2022-03-02T15:12:17.000Z"
}
```


(*)POST : /api/event/

Créé un évènement et attribue la propriété de l'évènement. Si l'événement est un RDV, envoie un mail à l'administrateur l'informant d'une demande de RDV

Entrées : category, name, date

Sorties : Tableau d'objets Json représentant les évènements créés
(401 (unauthorized) 422 (err validation) http 201 OK, 500 erreur interne)

(*)PUT : /api/event/rdv/:id

Utilisé par l'administrateur pour valider un RDV. Un mail est envoyé au demandeur pour l'informer de l'état de prise en compte de sa demande

Entrées :
id : id de l'événement
user_id : fournit par le token
is_valid : boolean, représentant la validation d'un RDV

Sortie :
Objet JSON avec les informations (id event, id user, is valid) de l'évènement
200 : OK, 404 événement non trouvé, 500 erreur interne,422 erreur de validation

(*)DELETE : /api/event/:id

Supprime un événement

Entrées :
id : id de l'événement

Sortie :
204 : OK, 404 événement non trouvé, 500 erreur interne,401 si non autorisé

Catégories des évènements : /api/event/category/

GET : /api/event/category/

sorties : tableau représentant les catégories d'évènement
http 200 OK, 500 erreur

```
[
  {
    "id_category": 7,
    "category_name": "Podcast "
  }
]
```


GET : /api/event/category/:id

Entrée : id de la catégorie
Sortie : objet représentant une catégorie
http 200 OK, 404 catégorie non trouvée, 500 erreur interne


```
 {
    "id_category": 7,
    "category_name": "Podcast "
  }
```


(*)POST : /api/event/category/

Entrée : category_name : nom de la catégorie
Sortie : http 201 créé, 422 erreur de paramètres d'entrées,
500 erreur interne, 204 création non possible

(*)PUT : /api/event/category/:id

Entrée :
category_name : nom de la catégorie,
id: id de la catégorie à modifier.
Sortie :
http 204 modifié, 422 erreur de paramètres d'entrées,
500 erreur interne, 404 modification non possible
403 si l'ID event est 1 (RDV), non modifiable

(*)DELETE : /api/event/category/:id

Entrée : id: id de la catégorie à supprimer.
Sortie : http 204 supprimé, 500 erreur interne, 404 suppression non possible 403 si l'événement est RDV(1)

Authentification : /api/session/

Get /api/session :

récupère les informations utilisateur
Entrées :
rien
Sortie :
Objet JSON représentant l'id user et son level
(retour 401 : unauthorised, 200 OK)


```
{
   "userId": 11,
   "userLevelString": "superadmin",
   "firstname": "Arnold",
   "lastname": "Twist"
}
```

CV : /api/cv/

(*)GET /api/cv/

Récupère les CVs d'un utilisateur

Entrée :

rien id utilisateur via le token

Sorties :

Les des CVs de l'utilisateur
HTTP 200 OK, 500 erreur, 404 pas de CVs


```
[
  {
    "id_cv": 1,
    "path": pat_to_cv",
    "id_user": 11,
    "is_owner": 1
  }
]
```


(*)GET /api/cv/admin/

Récupère les CVs pour l'administrateur

Entrée :

rien

Sorties :

Les des CVs
HTTP 200 OK, 500 erreur


```
[
  {
    "path": "path_to_cv",
    "id_cv": 1,
    "user": "Harry Stevenson",
    "id_users": 11
  }
]
```


(*)POST /api/cv/

Poste un CV

Entrée :

file : fichier du cv

Sortie :

HTTP 201 OK, 500 erreur, 422 erreur avec le fichier

(*)DELETE /api/cv/:id

Supprime un CV

Entrée :

id du CV

Sortie :

HTTP 204 OK, 404 CV non trouvé, 500 erreur interne

Entreprise : /api/enterprise/

GET /api/enterprise/

récupère la liste des entreprises pour administrateur connecté

Sorties

Tableau comprenant des objets JSON:

“name": nom de l’entreprise
"adress": adresse de l’entreprise
"id_activity": id de l’activité
"Nb_employes": nombre d’employés
(http 200 OK, 500 error)


```
[
  {
    "id_enterprise": 1,
    "name": "None",
    "adress": "adress",
    "id_activity": 15,
    "Nb_employes": 0
  }
]
```


GET /api/enterprise/:id

récupère les informations d’une entreprise par id pour administrateur connecté

Entrées

id : id de l’entreprise à récupérer

Sorties

Tableau comprenant des objets JSON:

“name": nom de l’entreprise
"adress": adresse de l’entreprise
"id_activity": id de l’activité
"Nb_employes": nombre d’employés
(http 200 OK, 500 error 404 : entreprise not found)


```
{
  "id_enterprise": 1,
  "name": "None",
  "adress": "adress",
  "id_activity": 15,
  "Nb_employes": 0
}
```


POST /api/enterprise/

Enregistre une entreprise pour un user connecté

Entrées

“name": nom de l’entreprise
"adress": adresse de l’entreprise
"id_activity": id de l’activité
"Nb_employes": nombre d’employés

Sorties

Tableau comprenant des objets JSON:

“name": nom de l’entreprise
"adress": adresse de l’entreprise
"id_activity": id de l’activité
"Nb_employes": nombre d’employés
(http 201 id_enterprise, 500 error 422 : données invalides)

PUT /api/enterprise/:id

Modifie une ou plusieurs entrées d’une entreprise par id pour administrateur connecté

Entrées

id : id de l’entreprise à modifier
“name": nom de l’entreprise
"adress": adresse de l’entreprise
"id_activity": id de l’activité
"Nb_employes": nombre d’employés

Sorties

Si entreprise non trouvé : erreur 404 'Enterprise not found'
Si données invalide : erreur 422 avec le détail
Si succès : 200 `Enterprise id= updated`

DELETE /api/enterprise/:id

Supprime une entreprise par id pour administrateur connecté

Entrées

id : id de l’entreprise à supprimer

Sorties

Information si supprimé : 200’'Enterprise deleted'
Erreur 404 si entreprise non trouvé ou autre : 500 'Error deleting enterprise'

Offres d'emploi : /api/job/

GET /api/job/

Récupère les offres d'emploi publiées sur le site

Entrées :

Rien

Sortie :

tableau d'objet JSON représentant les offres
HTTP 200 OK, 500 erreur interne


```
[
  {
    "id_job": 10,
    "name": "Test Offre4",
    "path": "uploads\\offers\\25-2-2022_7404.pdf",
    "name_offer": "Indépendant",
    "category_name": "Agriculture, agroalimentaire",
    "id_type": 6,
    "cat_job": 1
  }
]
```


(*)POST /api/job/

Envoi une offre d'emploi :

Entrées :

file : fichier offre d'emploi
id_user : utilisateur ayant posté l'offre (pour le moment seul un super admin peut poster, via le token)
id_type : entier représentant l'id du type de contrat
name : nom de l'offre

Sortie :

HTTP 201 OK, 422 erreur avec les données, 500 erreur interne

(*)DELETE /api/job/:id

Envoi une offre d'emploi :

Entrées :

id : entier représentant l'id de l'offre

Sortie :

HTTP 204 OK, 500 erreur interne,404 offre non trouvée

GET /api/job/offertype/

Récupère le type d'offre d'emploi

Entrées :

Rien

Sortie :

tableau d'objet JSON représentant les types d'offre d'offre
HTTP 200 OK, 500 erreur interne

GET /api/job/count/

Récupère le nombre d'offre d'emploi

Entrées :

Rien

Sortie :

Objet JSON représentant le nombre d'offre d'offre
HTTP 200 OK, 500 erreur interne


```
{
  "nb": 7
}
```

(*)GET /api/job/admin/

Récupère les offres d'emploi publiées sur le site pour l'administrateur

Entrées :

Rien

Sortie :

tableau d'objet JSON représentant les offres
HTTP 200 OK, 500 erreur interne


```
[
  {
    "id_job": 10,
    "name": "Test Offre4",
    "path": "uploads\\offers\\25-2-2022_7404.pdf",
    "name_offer": "Indépendant",
    "category_name": "Agriculture, agroalimentaire"
  }
]
```


GET /api/job/category/

Récupère les catégories d'offres d'emploi publiées sur le site

Entrées :

Rien

Sortie :

tableau d'objet JSON représentant les catégories d'offres
HTTP 200 OK, 500 erreur interne


```
[
  {
    "id_job_category": 1,
    "name": "Agriculture, agroalimentaire"
  },
]
```

Avis : /api/opinion/

(*) GET /api/opinion/

Récupère la liste des avis, pour le panneau d'administration

Sortie :

Tableau comprenant des objets JSON:
opinion : le texte de l'avis
author : auteur de l'avis au format prénom nom
id_opinion : l'id de l'avis
is_valid : booléen permettant de changer la visibilité de l'avis
(http 200 OK, 500 error)


```
[
  {
    "id_opinion": 5,
    "is_valid": 1,
    "opinion": "Très bonne entreprise",
    "author": "Harry Twist"
  }
]
```


GET /api/opinion/eurheka/

Retourne les avis concernant la société Eurhéka.

Sortie :

Tableau comprenant des objets JSON:
opinion : le texte de l'avis
author : auteur de l'avis au format prénom nom
(http 200 OK, 500 : error)


```
[
  {
    "id_opinion": 5,
    "is_valid": 1,
    "opinion": "Très bonne entreprise",
    "author": "Harry Twist"
  }
]
```


(*) GET /api/opinion/:id

Récupère un avis en fonction de l'id

Entrées :

id : id de l'avis à récupérer

Sortie :

Un objet JON:
opinion : le texte de l'avis
author : auteur de l'avis au format prénom nom
id_opinion : l'id de l'avis
is_valid : is_valid : booléen permettant de changer la visibilité de l'avis
(http 200 : OK, 404 : not found, 500 : error)


```
{
    "id_opinion": 5,
    "is_valid": 1,
    "opinion": "Très bonne entreprise",
    "author": "Harry Twist"
  }
```

(*) GET /api/opinion/enterprise/:id

Récupère les avis sur une entreprise

Entrées :

id : id de l'entreprise

Sortie :

Un tableau d'objets JON:
opinion : le texte de l'avis
author : auteur de l'avis au format prénom nom
(http 200 OK, 500 error, 404 : entreprise not found)

(*)POST /api/opinion/

Enregistre un avis sur une entreprise

Entrées

-enterprise* : int, représente l'entreprise (optionnel, si manquant sera remplacé par l'ID de Eurheka

-opinion : text, le texte de l'avis

Sortie

Objet JSON avec le nouvel avis
(http 201 OK, 500 : error, 422 : données invalides)

(*) PUT /api/opinion/:id

Modifie la visibilité d'un avis

Entrées

-id : id de l'avis
-is_valid : booléen représentant la visibilité de l'avis

Sorties

Objet JSON avec id de l'avis et état actuel (chaine de caractère si rien de fait)
(http 200 OK, 404 avis non trouvé, 422 : données invalides, 500 :erreur, 204 si rien de modifié)

(*) DELETE /api/opinion/:id

Supprime un avis

Entrée :

id de l'avis à supprimer

Sortie :

Information si bien supprimée
(http 200 OK, 500 error)

Thèmes : /api/theme/

GET /api/theme/

Récupère les thèmes

Sorties : tableau d'objets représentant les thèmes

(http 200 OK, 500 erreur)

```
[
 {
      "id_theme": 22,
      "name": "Carrière"
   }
]
```

GET /api/theme/:id

Récupère un thème par son ID

Entrées :

ID : Id du thème

Sortie :

Objet JSON représentant le thème

HTTP 200 si trouvé, 500 si erreur

```
 {
      "id_theme": 22,
      "name": "Carrière"
   }
```

(*)POST /api/theme/

Ajoute un thème

Entrées :

name : Nom du thème

Sortie :

HTTP 201 si créé,422 si erreur de saisie ,500 si erreur

(*)PUT /api/theme/:id

Modifie un thème par son ID

Entrées :

ID : Id du thème

name : nouveau nom

Sortie :

HTTP 200 si modifié, 422 si erreur de saisie, 500 si erreur

(*)DELETE /api/theme/:id

Supprime un thème par son ID

Entrées :

ID : Id du thème

Sortie :

HTTP 200 si supprimé,404 si élément non trouvé, 500 si erreur

GET /api/theme/admin/

Récupère les thèmes

Sorties : tableau d'objets représentant les thèmes et une information de coche, pour la page admin (http 200 OK, 500 erreur)


```
[
  {
    "id_theme": 22,
    "name": "Carrière"
  }
]
```

Utilisateurs : /api/users/

(*)GET /api/users/admin/

Récupère la liste des utilisateurs pour le panel admin

Sorties :

Tableau JSON représentant l'id, le nom et prénom, l'entreprise et le level des utilisateurs
http 200 OK, 500 erreur interne


```
[
  {
    "id_users": 13,
    "userName": "Durant Marie",
    "user_level": 2,
    "name": null
  }
]
```


(*)GET /api/users/:id

Récupère les informations d'un utilisateur. Retourne 403 si le demandeur n'est pas l'utilisateur ou un super admin

Entrées :

id : Id de l'utilisateur

Sorties :

Objet JSON représentant les informations utilisateurs
HTTP 200 OK, 403 interdit, 404 non trouvé


```
{
  "id_users": 13,
  "firstname": "Marie",
  "lastname": "Durant",
  "email": "marie@gmail.com",
  "phone": "0000000000000",
  "birthday": null,
  "adresse": "Ma belle adresse",
  "in_post": 0,
  "free_date": null,
  "job_search": 0,
  "job_name": null,
  "job_date": null,
  "enterprise_name": null,
  "signin_options": 1
}
```


POST /api/users/

Enregistre un utilisateur basique

Entrées

-firstname : string, prénom
-lastname : string, nom
-email : email de l'utilisateur, unique
-password : string, mot de passe, min 8 caractères, max 50
-focus : booléen, optionnel : faire le point
-accompanied : booléen, optionnel : demande d'accompagnement
-stage booléen, optionnel, demande de stage ou d'emploi

Sortie

Objet JSON avec l'id créé
(http 201 OK, 500 : error, 422 : données invalides, 409 : utilisateur existant)

(*)PUT /api/users/:id

Mets à jour les informations d'un utilisateur

Entrées :

email: email
firstname:prénom
lastname: nom
password: mot de passe
adresse: adresse civile
phone: numéro de téléphone
birthday: date de naissance
in_post: Actuellement en poste
free_date: Date de fin de mission ou de disponibilité
enterprise_name: Nom de l'entreprise
job_date: date d'embauche
job_name: nom du poste occupé
job_search: A la recherche d'un emploi booléen

Sorties :

204 : ok, 404 utilisateur non trouvé, 422 : erreur de données, 500 erreur interne

(*)PUT /api/users/admin/:id

Modifie le niveau d'utilisateur

Entrées :

-id de l'utilisateur
- user_level : nouveau niveau de l'utilisateur

Sorties :

204 : OK, 404 utilisateur non trouvés, 500 erreur interne

(*)DELETE /api/users/:id

Supprime un utilisateur

Entrées :

-id :id de l'utilisateur à supprimer

Sorties :

HTTP 204 : OK, 500 Erreur avec traitement de l'erreur de cascade, 404 user non trouvé.

POST /api/users/login/

Permet la connexion d'un utilisateur

Entrées :

-email, string
-password, string entre 8 et 50 caractères

Sortie :

Objet Json + Cookie jwt avec le token.
(http 200 OK, 500 error, 422 données invalides, 401 : accès refusé, mot de passe invalide)


```
{
  "userId": 11,
  "firstname": "Harry",
  "lastname": "Twist"
}
```


GET /api/users/logout/

Permet de se déconnecter de l'interface

Entrées :

rien

Sortie :

suppression du cookie de token, renvoie à /

POST /api/users/lost/

Route utilisée lorsqu'un utilisateur demande une réinitialisation de son mot de passe. Envoie un mail à l'utilisateur avec un token pour changer son mot de passe

Entrées :

-email, string

Sortie :

(http 204 OK, 500 error, 422 données invalides, 404 : usager non trouvé)

POST /api/users/newPass/

Permet une modification du mot de passe lors d'une perte de celui-ci

Entrées :

-email, string
-password : string
-token : string

Sortie :

(http 204 OK, 500 error, 422 données invalides, 404 : usager non trouvé)

Envoi d’email : /api/mail/

POST /api/mail/

Envoyer un email depuis le formulaire de contact

Entrées :

- subject :string, sujet
- message : string, corps du message
- email: string, adresse mail de l’expéditeur
- lastname : string, nom de l’expéditeur
- firstname : string, prénom de l’expéditeur

Sorties :

- http 200 OK, 500 : erreur, 422 erreurs de champs